09/752,026.

L Number	Hits	Search Text	DB	Time stamp
1	6653	(707/1 707/10 707/104.1 707/100).ccls.	USPAT	2004/07/13 14:49
2	1	identifi$6 same device same application$1 same user same regist$6 same timestamp	USPAT	2004/07/13 14:55
3	867	715/513.ccls.	USPAT	2004/07/13 14:55
4	7247	715/513.ccls. ((707/1 707/10 707/104.1 707/100).ccls.)	USPAT	2004/07/13 14:55
5	822	identifi$6 same device same application$1 same regist$6	USPAT	2004/07/13 14:56
6	92612	identifi$6 same device	USPAT	2004/07/13 14:56
7	21	(715/513.ccls. ((707/1 707/10 707/104.1 707/100).ccls.)) and (identifi$6 same device same application$1 same regist$6)	USPAT	2004/07/13 15:07
8	5	web same page$1 same location same wildcard	USPAT	2004/07/13 15:12

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Tuesday, July 13, 2004

Hide?	Set Name	Query	Hit Count
		DB=PGPB; PLUR=NO; OP=ADJ	
☐	L6	L5 and l4	7
☐	L5	((web adj1 page$1) or HTML)	25894
☐	L4	L3 and l2	13
☐	L3	(707/1 or 707/100 or 707/10 or 707/104.1 or 715/513).ccls.	4881
☐	L2	storing same retrieving same variable$1	128
☐	L1	(707/1 715/513 707/100 707/10 707/104.1).ccls.	0

END OF SEARCH HISTORY

WEST Search History

Hide Items | Restore | Clear | Cancel

DATE: Tuesday, July 13, 2004

Hide?	Set Name	Query	Hit Count
	DB=USPT; PLUR=NO; OP=ADJ		
☐	L9	L8 and l7	11
☐	L8	707/$.ccls.	13241
☐	L7	location same homepage	49
	DB=PGPB; PLUR=NO; OP=ADJ		
☐	L6	L5 and l4	7
☐	L5	((web adj1 page$1) or HTML)	25894
☐	L4	L3 and l2	13
☐	L3	(707/1 or 707/100 or 707/10 or 707/104.1 or 715/513).ccls.	4881
☐	L2	storing same retrieving same variable$1	128
☐	L1	(707/1 715/513 707/100 707/10 707/104.1).ccls.	0

END OF SEARCH HISTORY

L6: Entry 4 of 7 File: PGPB Apr 25, 2002

DOCUMENT-IDENTIFIER: US 20020049788 A1
TITLE: Method and apparatus for a web content platform

Current US Classification, US Primary Class/Subclass:
715/513

Summary of Invention Paragraph:
[0014] For example, prior art web content development systems use HTML and Java code in the same file. The web content is mostly based on HTML, but HTML does not separate the information from its presentation, mixing formatting tags, descriptive tags and programmable logic. Typically the page development process starts with an HTML mockup of the page. In the development process, the HTML page was then modified by a software engineer to include scriptlets (invoking bean methods, extracting parameters, internationalization, etc.). Once the page was complete, it was relatively complicated to change the logic or the look of the page, since both aspects of the page reside in the same page as a jumbled mix of scriptlets and HTML markup.

Detail Description Paragraph:
[0057] //developerjava.sun.com/developer/earlyAccess/j2sdkee/doc-beta/guides/ejb/html/TOC.html

Detail Description Paragraph:
[0076] WDK (Web Development Kit) server 523 is Saba's web content generation engine. Using web standards for XML and XSL, it provides a customizable framework for decoupling data from presentation, and generating web content in a variety of formats, from standard HTML to WML. The WDK 523 provides the following base interfaces:

Detail Description Paragraph:
[0217] Referring now to FIG. 4, the tier 3 applications server 307 is expanded in FIG. 4 to illustrate the Business Applications Platform 415 of the present invention. In FIG. 4, the Platform contains an Interface Server 417, an Information Server 419, an Interconnect Server 423 and a Business Server 421. All of these Servers 417, 419, 421 and 423 may physically reside on the same hardware platform (such as a UNIX box or a Microsoft.TM. NT.TM. platform), or each server may reside on a separate hardware box, or any combination of servers and hardware boxes. Each of the servers may have included a JAVA Virtual Machine.TM. and the related runtime support. The electronic communications between these servers may use the XML protocol (409, 425, 427) with each server having services for translating XML into the particular Applications Programming Interface (API) language required by the server and for translating its internal language into XML prior to transmission to another server. In a preferred embodiment, all of these servers are contained in a single tier 3 platform, and may communicate with each other directly without the necessity of changing the interfacing protocol format. The Interface Server 417 (also alternatively designated herein as the WDK), communicates through a web server 405 via the internet 403 to web clients 401 via the HTML protocol. The Interface Server 417, also may communicate to a directly connected client 407 via other protocols such as XSL/XSLT etc., and may communicate to Personal Data Assistants 411 such as cell phones or Palm Pilots.TM. or other such wireless

devices using wireless protocols such as WAP/WML, etc. The Interface Server 417, contains mechanisms to manipulate various kinds of display style sheets, to generate and execute web links, to manage dynamic content generation and dynamic generation of Javascript, all of which is described in more detail below in the section on the Interface Server/WDK 417.

Detail Description Paragraph:
[0348] In the alternative embodiment, some of the metadata that is currently captured about a class or an attribute could be dynamically determined using the Java reflection API. Examples include the parent ID and attribute count for business objects and attribute type for an attribute. The Java reflection API provides classes Class and Field that can be used to retrieve such information. Furthermore, instead of building a hashtable-based infrastructure for storing and retrieving attribute values, one can use methods like set and get in the Field class to operate directly on the attributes, which are declared as member variables of the class.

Detail Description Paragraph:
[0519] Web Content Server 800 can also provide the platform's web content generation engine for use by users to create, render, and present web content while improving the dynamic acquisition of data from a variety of sources followed by its reformatting and display via style sheets. Using web standards for XML and XSL, Web Content Server 800 provides a user with a customizable framework for decoupling data from presentation, and generating web content in a variety of formats, from standard HTML to WML.

Detail Description Paragraph:
[0531] The platform 808 allows content, logic and style to be separated out into different XML files, and uses XSL transformation capabilities to merge them resulting in the automatic creation of H.TM.L through the processing of statically or dynamically generated XML files. The platform 808 can also generate other, non-HTML based forms of XML content, such as XSL:FO rendering to PDF files, client-dependent transformations such as WML-formatting for WAP-enabled devices, or direct XML serving to XML and XSL aware clients.

Detail Description Paragraph:
[0532] The platform 808 divides the development of web content into three separate levels: (a) XML creation-The XML file is created by the content owners. They do not require specific knowledge on how the XML content is further processed-they only need to know about the particular chosen "DTD" or tagset for their stage in the process. This layer can be performed by users directly, through normal teeditors or XML-aware tools/editors; (b) XML processing-The requested XML file is processed and the logic contained in its logicsheet is applied. Unlike other dynamic content generators, the logic is separated from the content file; and (c) XSL rendering-The created document is then rendered by applying an XSL stylesheet to it and formatting it to the specified resource type (HTML, PDF, XML, WML, XHTML, etc.).

Detail Description Paragraph:
[0535] Using platform 808, developing a web page (web content) requires the development of the following components: (a) a control file; (b) a model file; (c) a view file; and (d) Command Managers and Commands.

Detail Description Paragraph:
[0539] The View contains all style and presentation for a given page. Users are responsible for implementing an XSLT stylesheet that transforms the model into a specific presentation environment. View developers are typically UI designers, since the bulk of authoring effort is crafting the HTML for a static page, then adding in the set of XSLT tags to create a stylesheet for the associated model page.

Detail Description Paragraph:
[0543] The process of creating the HTML to send to the browser begins with reading the control file, 860. The control file 862 is simply a file that identifies the model file 864, the view file 866 and the widget library 868 to use to produce the final HTML result 870. The control file 862 also contains link transformation information that is used to transform links used in the model file 864. This link transformation is used to map model-file hyperlink references contained in the model file 864 to appropriate control file names.

Detail Description Paragraph:
[0545] The process outlined above also highlights how the different aspects of developing dynamic web content are separated. The design of a particular web page is the result of answering the following questions: (a) What do I do with parameters sent from the browser and what data is needed to display the page? How do I perform these tasks? (b) How will the user interact with the page? What buttons, entry fields etc. will the user have? and (c) How are the data and the interaction elements displayed on the page?

Detail Description Paragraph:
[0548] Typically the page development process starts with an HTML mockup of the page. The Web Content Server 800 development process can start with the HTML mockup as well. However, users do not modify this mockup to include code. Instead the process illustrated in FIG. 8C is followed.

Detail Description Paragraph:
[0549] As illustrated in FIG. 8C, using the HTML mockup 884, the user develops three specifications. The data model specification 886 is developed to meet three basic criteria. First, the data model needs to contain enough information to drive the interface. For example, if the interface needs to display the name of an object, then the data model must contain the object name in some form. Second, the data model specification should maximize reuse of command objects. For example, if a command object already exists that can retrieve a needed object in a serialized XML format, then the data model of the command object should be reused instead of reinventing a new XML representation of the same object. Finally, the data model specification should be generic so other pages can reuse the model generation components (Commands). How general the data model should be is determined by balancing the trade-off between performance (since producing more data may incur performance penalty) and reusability. If producing a more general data model causes high performance penalty, then a less general solution may be better. On the other hand, if adding a few not needed items comes at no or little performance cost, then the more general data model is preferred. For example, objects implementing the IXMLObject interface will typically provide more than enough information about themselves. The data model specification 886 should essentially be a sample of the data returned by the Command objects and the specification XML should be wrapped in tags.

Detail Description Paragraph:
[0552] Once the specifications 886, 888, and 890 are complete, the user or a tool, produces a sample model instance 892. The user can use the model instance 892 to test the view stylesheet (by using any standard XSLT tool). The user develops the view stylesheet by converting the original HTML mockup to an XSLT stylesheet to retrieve dynamic data, widgets and internationalized labels from the model instance. This conversion process can mostly be done in an HTML editor.

Detail Description Paragraph:
[0556] To change the look and feel of textual and graphical information, the user can edit the view page in an HTML tool. The user can add , etc. tags around the components needed modification, and define the "style" attribute to reflect the desired look and feel changes. If the user needs to develop for browsers with limited CSS support (e.g., Netscape 4.x), the user can wrap the

components in <u>, <b>, <font>, etc. tags as needed.

Detail Description Paragraph:
[0558] The cut/copy/paste commands of the HTML editor can be used to perform most layout changes requiring the repositioning of different components. Dreamweaver, for example, gives users powerful HTML/XML element selection capabilities that make it easier to move and copy whole HTML/XML document fragments.

Detail Description Paragraph:
[0560] Often the model specification will result in the production of more content than needed by a particular view. For example, the model for a page that needs to display the parents of a particular security domain only may also produce other information about the security domain (e.g., the description of the domain). This is especially likely when the model page reuses other, already existing command objects. In such cases displaying additional content can simply be done at the view page level: the user needs to place the newly required information somewhere on the view page. Removing information items is also very simple, since users can simply delete a particular HTML/XML fragment if viewing that piece of the model is not needed.

Detail Description Paragraph:
[0569] The view page displays the data and widgets contained in the model instance (i.e. the XML document produced by executing the model page). If the control page declares a widget library to use, then the view transformation takes place after the widgets have already been transformed to the appropriate format (e.g. HTML).

Detail Description Paragraph:
[0570] The widget library contains the display transformation for widget components. After the model page executes the produced widgets are transformed to the appropriate output format (e.g., HTML). The resulting HTML markup is wrapped in tags so the view transformation page can easily identify and place each widget.

Detail Description Paragraph:
[0581] In one preferred embodiment, the Web Content Server 800 is a dynamic content generation framework based on the apache Cocoon project. Like other approaches, such as JSP, ASP, ColdFusion etc., the Web Content Server 800 would allow developers to create web pages to display data derived dynamically through some business logic. Unlike other dynamic content generation frameworks, the Web Content Server 800 separates the content from its presentation. This separation makes it easier to customize pages, to provide different versions of pages to different user agents (desktop browsers, handheld devices, etc.).

Detail Description Paragraph:
[0673] The default XSLT processor that comes with Cocoon performs a single XSLT transformation only. However, Web Content Server 800 requires two XSL transformations after the java code produces the data. The first transformation replaces the widgets with their HTML representation (the widget transformation) while the second transformation renders the data (the view transformation). To make the engine aware of the Web Content Server 800 XSLT processor, the following line is added to the cocoon.properties file:

Detail Description Paragraph:
[0695] wdk:form-The wdk:form element is one of the elements in the widget library. Since most wdk pages are HTML forms, the wdk:form element is used to generate the HTML form and javascript functions required by a Web Content Server 800 application. For example, a javascript function is generated that can be called by link widgets to submit the form..

Detail Description Paragraph:
[0711] View pages are XSLT stylesheets. The role of the view stylesheet is to

convert the XML document produced by executing the model file (and the subsequent widget transformation) to a format understood by the user agent. For example, for desktop browsers this typically means conversion to an HTML representation. Since model pages have a well-defined structure, view pages are also highly regular. For example, there are a number of model page elements that should not be rendered (such as wdk:head element and its content should not be copied to the output). Other model pages nodes have a standard representation in HTML (or in the desired output format). For example, the rule for rendering wdk:page is to generate the <html> element, the <head> element containing the <title> element. These common templates are all grouped in a default stylesheet that can be imported using the <xsl:import> directive by every view page. As a result, for simple pages, the view page needs to contain a singe customized xsl:template rule that matches on the "wdk:model" node. This template is responsible for rendering the data as well as the widgets.

Detail Description Paragraph:
[0713] Section 1 defines the namespaces used in the stylesheet. Section 2 defines the root level template. This template produces the html tags, and generates the html head element complete with the title element. Section 3 defines the default template: every element, attribute, text and comment is copied to the resulting document, unless a more specific template provides different instructions. Section 4 specifies a template for eliminating the wdk:head and wdk:widgets elements and their contents (since the contents of these tags should not be rendered using the default template defined in section 3). Section 5 introduces a template for transforming every widget by wrapping them into a span element replacing the wdk:widget "wrapper". This makes it possible to use CSS styling on a per named-widget basis. Finally, section 6 defines the template for processing the wdk:page element.

Detail Description Paragraph:
[0740] The Web Content Server 800 widget library contains rules (XSLT templates) for transforming a number of widgets to their HTML representation.

Detail Description Paragraph:
[0741] The widget library provides a level of abstraction between the user interaction component (e.g., a text input field) and its presentation (e.g., an HTML input field or a WML input field). This way the content producing model pages can be reused by different control files-one may deliver the content to a desktop browser using the HTML widget library, while another may deliver the same content to a handheld device using a modified version of the widget library (e.g., using WML).

Detail Description Paragraph:
[0751] Section 1 contains the match condition for the template: every wdk:input element in the document will be transformed using this template. In section 1 the name of the input field is computed as well. Section 2 shows that this widget Oust like all the other widgets) is nested inside a wdk:widget element, which makes it simpler to place widgets in the view transform. Section 3 shows how the different components (the label and the actual text field) are embedded in an HTML span element. In section 4 the color of the text label is determined based on the "required" sub-element of the wdk:input widget. The logic in section 5 determines what type of text field to generate: either "password" or regular "text" field. Section 7 shows the template called from section 5 to fill in the attributes of the generated HTML input element.

Detail Description Paragraph:
[0753] wdk:hidden_element: Represents an HTML hidden element. The widget generates the required element and Javascript functions that can be invoked to set the value of this element.

Detail Description Paragraph:
[0754] wdk:form: Generates the HTML form element and Javascript functions needed to manage the form.

Detail Description Paragraph:
[0755] wdk:input: Represents a single line text element. Can render the widget as a PASSWORD or TEXT HTML form field.

Detail Description Paragraph:
[0756] wdk:list: Represents a widget for selecting an item from a set of predefined items. Supports four different HTML renderings:

Detail Description Paragraph:
[0925] In an embodiment of the invention, Information Distributor 1201 provides a flexible mechanism for annotating and matching web resources 1200. Information Distributor 1201 can locate and deliver a wide variety of resources, from web pages to Business Objects. Information Distributor 1201 also supports a wide variety of descriptive information required by business applications, from standard web metadata to catalog information to skills and competencies.

Detail Description Paragraph:
[0938] Import Agents 1300 create and import the RDF descriptions used by IDK. Import Agents 1300 can generate metadata from a variety of sources, from existing web pages and business objects to content management systems to enterprise applications.

Detail Description Paragraph:
[0940] Delivery Agents 1304 dispatch the results of a query or match. In an embodiment, Delivery Agents 1304 integrate with a variety of delivery mechanisms, from web page generation and XML datagrams to email and event messaging systems.

Detail Description Paragraph:
[1153] In a preferred embodiment, the business server 1727 embodies the containers which incorporate all of the business logic, common business objects, SABA core objects, and a database driven framework for generating notifications and for triggering periodic events based on context sensitive attachments. The business server 1727 communicates with each of the other servers within the Platform using the XML protocol (1727, 1729, and 1731). The Business Server 1727 also communicates with the database management system 1713. In communicating with the interface server 1721, the business server 1727 first generates a XML message 1729 and transmits it to the interface server 1721. The interface server 1721 then performs style sheet transformations on the XML using XSL or XSLT to translate the XML message into the particular Applications Programming Interface (API) language required to communicate with a particular user. For example, if a particular user is accessing the Platform via a workstation 1701 or a PC 1703, the Interface Server 1721 can convert the XML 1729 into HTML 1735 and communicate with the user through a web server 1707 via the Internet 1709. The Interface Server 1721 can also convert the XML into other protocols such as WAP/WML 1737 to communicate with Personal Data Assistants (PDAs) such as cell phones 1705, Palm Pilots.TM., or other such wireless devices. Since the interface that is generated between the Platform and the various user interfaces is dictated by the set of style sheets generated in the Interface Server 1721, the same core business logic can be leveraged to communicate across a number of different user interfaces.

Detail Description Table CWU:
108 <?xml version="1.0"?> <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax- ns#" xmlns:dc="http://purl.org/dc/elements/1.1/" xmlns:schedule="http://www.saba.com/RDF/schedule/1.0#"> <rdf:Description resource="http://dlipkin/class1"> <dc:title>HTML Fundamentals</dc:title> <schedule:startDate>1998-12-07</schedule:startDate> </rdf:Description> </rdf:RDF>

Detail Description Table CWU:
109 rdf_resource rdf_property rdf_object http://dlipkin/class 1
http://purl.org/dc/elements/1.1/title HTML Fundamentals http://dlipkin/class 1
http://www.saba.com/RDF/sched- ule/1.0#startDate 1998-12-07

Previous Doc Next Doc Go to Doc#